Software Tech News

The DoD Source for Software Technology Information.

Vol. 4- No. 1
High Performance Computing

In This Issue:

www.dacs.dtic.mil/awareness/newsletters/listing.shtml

Air Force Research Laboratory - Fielding the Fastest Embeddable Computer in the Air Force

by Virginia Ross, AFRL - Rome Research Site

Our Mission: Leading the discovery, development, and integration of affordable, warfighting technologies for our aerospace forces.

The Air Force Research Laboratory in Rome, New York is purchasing a rugged COTS High Performance Computer (HPC) system from SKYComputers of Chelmsford, Massachusetts. A unique attribute of this ruggedized HPC is its availability to DoD RDT&E organizations and their contractors to accelerate the transition of new HPC applications to field use. Some potential uses include: advanced signal and image processing research in sensor and information fusion, synthetic aperture radar, space-time adaptive processing, automatic target recognition, wavelet-based compression, and hyper-spectral imaging. This system is the fastest embeddable computer in the U.S. Air Force. It was funded through the DoD's High Performance Computing Modernization Program (HPCMP), representing the first purchase of a rugged HPC system by the program.

Continued on page 17

DoD Data & Analysis Center for Software
http://www.dacs.dtic.mil

High Performance HPC Computing

Graph Partitioning for Load Balancing of Multi-phase, Multi-physics, and Multi-mesh Simulations

by Kirk Schloegel, George Karypis, VipinKumar, Army HPC Research Center

Introduction

Algorithms that find good partitionings of irregular graphs are critical for the efficient execution of scientific simulations on high-performance parallel computers. In these simulations, computation is performed iteratively on each element (and/or node) of a physical two- or three-dimensional mesh and then information is exchanged between adjacent mesh elements. For example, computation is performed on each triangle of the two-dimensional mesh shown in Figure 1. Then information is exchanged for every face between adjacent triangles. The efficient execution of such simulations on parallel machines requires a mapping of the computational mesh onto the processors such that each processor gets roughly an equal number of mesh elements and that the amount of inter-processor communication required to perform the information exchange between adjacent elements is minimized. Such a mapping is commonly found by solving a graph partitioning problem [9]. For example, a graph partitioning algorithm was used to decompose the mesh in Figure 1. Here, the mesh elements have been colored to indicate the processor to which they have been mapped.

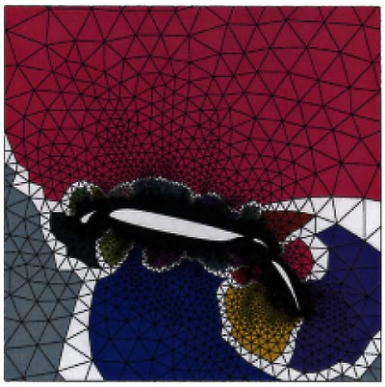


Figure 1: A partitioned 2D irregular mesh of an airfoil.

The shading of a mesh element indicates the processor to which it is mapped.

The graph partitioning problem is known to be NP-complete. Therefore, it is not possible to compute optimal partitionings for graphs of interesting size in a reasonable amount of time. This fact, combined with the importance of the problem, has led to the development of several heuristic approaches. See [9] for a recent survey of these schemes. Of these, multilevel algorithms are widely recognized as the state-of-the-art, as they are able to robustly compute high-quality partitionings quickly. Furthermore, many of these are available as serial (Chaco [2], JOSTLE [13], and MUDS [4]) or parallel (ParMeTiS [6] and PJOSTLE [12]) software libraries.

While graph partitioning algorithms have enabled the efficient execution of a wide range of scientific simulations on parallel machines, other applications have a number of additional requirements for their mesh decompositions that traditional partitioners are unable to satisfy. For example, many scientific simulations consist of a number of computational phases separated by synchronization steps (i.e., *multi-phase* simulations). These require that each of the phases be individually load balanced. Still other scientific simulations model multiple physical phenomenon (i.e., *multi-physics* simulations) or employ multiple meshes simultaneously (i.e., *multi-mesh* simulations). These also impose additional requirements that the partitioning algorithm must take into account. In this article, we describe some of these classes of simulations, as well as highlight new, generalized partitioning formulations and algorithms designed for them.

Multi-phase Simulations

Multi-phase simulations consist of a number of distinct computational phases, each separated by an explicit synchronization step. In general, the amount of computation performed for each element of the mesh is different for different phases. The existence of the synchronization steps between the phases requires that each phase be individually load balanced. That is, it is not sufficient to simply sum up the relative times required for each phase and to compute a decomposition based on this sum. Doing

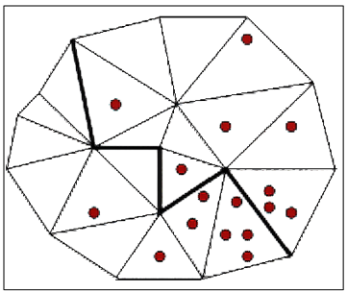


Figure 2: A particle-in-cell computation.

Here, both the mesh nodes and the particles must be distributed equally across the processors.

so may lead to some processors having too much work during one phase of the computation (and so, these may still be working after other processors are idle), and not enough work during other phases (and so these may be idle while other processors are still working). Instead, it is critical that every processor have an equal amount of work from all of the phases of the computation. A traditional graph partitioning scheme can be used to balance the load across the processors for a single phase of the computation. However, the load may be seriously imbalanced for the other phases. Another method is to use a different partitioning for every phase, each of which balances the load of a single phase only. This method requires that costly data redistribution be performed after each phase in order to realize the partitioning corresponding to the next phase. A better method is to compute a single partitioning that simultaneously balances the work performed in each of the phases. In this case, no redistribution of the data is necessary, and all of the phases are well balanced.

Figures 2 and 3 give examples of multi-phase simulations. Figure 2 shows a mesh and the particles from a particle-in-mesh simulation. This computation is composed of two phases. The first phase is the mesh based computation, and the second phase is the particle-based computation. In order to load balance such a simulation, each processor must have a roughly equal amount of both the mesh computation and the particle computation. This is not trivial because the number of particles within each mesh element can be different. Therefore, while ensuring that each processor has an equal number of mesh elements will load balance the mesh-based computation, it does not guarantee that each processor has an equal number of particles. Likewise, ensuring that each processor has an equal number of particles does not guarantee that they also have equal numbers of mesh elements. The dark line in Figure 2 gives a single bisection that splits both the mesh elements and the particles evenly. Figure 3 illustrates the mesh associated with the simulation of the ports and the combustion chamber of an internal combustion engine. Here, the simulation is performed in six computational phases. (Each of these corresponds to a different color in the figure.) In order to solve such a multi-phase computation efficiently on a parallel machine, every processor should contain an equal number of mesh elements of all six different colors. Figure 4 shows two subdomains from an 8-way partitioning of the mesh in Figure 3. This partitioning (computed by the multi-constraint partitioning algorithm implemented in METIS[4]) balances all six of the phases while also

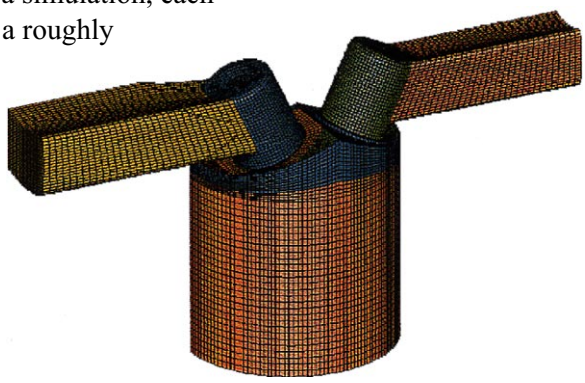


Figure 3: An internal combustion engine simulation

This figure is an example application whose computation is performed in multiple phases. Each color represents elements active during a different phase. (Figure provided by Analysis and Design Application Company Ltd.)

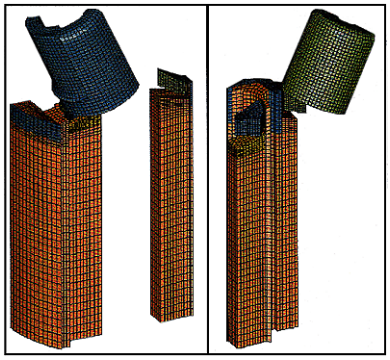


Figure 4: Two subdomains

Two subdomains of an 8-way partitioning computed by the multi-constraint graph partitioner implemented in MOOS 4.0 are shown. Note, that all of the subdomains have an equal number of elements of each color (although they're not all visible). (Figure provided by Analysis and Design Application Company Ltd.)

Continued on page 4

continued frommpage 3

minimizing the inter-processor communications. (Note that not all of the colors are visible in Figures 3 and 4.)

Multi-physics Simulations

Many computations simulate a variety of materials and/or physical phenomenon together. An example is the elastic-plastic soil-structure interaction computations that are used to simulate static and dynamic (earthquake) loading events. In such simulations, elastic computations are performed on each of the elements of the mesh. After these are completed, a yield condition is checked for every mesh element. This check determines whether or not the plastic computation must be performed on the element. Typically, zones in a three-dimensional solid may become plastic (i.e., load) and then elastic (i.e., unload). Thus, the extent of the plastic zone changes dynamically. This change can be both slow and rapid. Slow change usually occurs during initial loading phases, while the later deformation tends to localize in narrow zones rapidly [3]. (See Figure 5.) This is an example of a dynamically evolving computation that has multiple (i.e., two) phases. Due to its dynamic nature, not only is a static decomposition required, but periodic load balancing must also be performed to maximize efficiency.

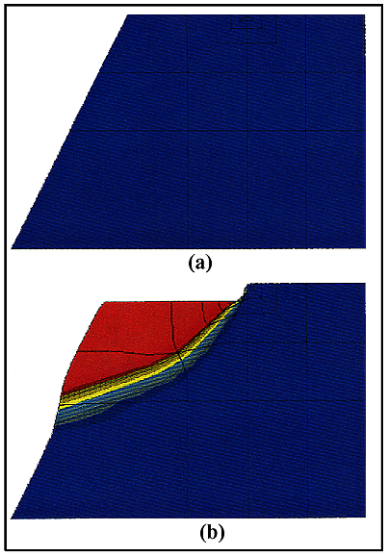


Figure 5: Two meshes associated with an earthquake simulation.

The initial mesh, in which an elastic computation is required for each mesh element, is shown in (a). The adapted mesh is shown in (b). Here, an elastic computation is still required for each mesh element. However, a plastic computation is also required for the mesh elements that are located on the narrow zone between the red and blue regions.

Multi-mesh Simulations

Another important class of emerging methods are multi-mesh computations. Multiple meshes arise in several settings that use grids to discretize partial differential equations. For example, some operations are innately more efficient on structured grids, such as radiation transport sweeps. However, complex geometries are better fitted with unstructured meshes. In some simulations, both kinds of grids may be used throughout the computation. Similarly, various codes that solve for multiple physical quantities (eg., multi-physics computations) may use separate grids to solve the appropriate equations for each variable. For example, consider a simulation of the welding of a joint between two parts, a process in which the parts are pressed together and thermally annealed [7]. One grid could be used for the solution of the stress-strain relations that mediate the mechanical deformation of the parts. A second grid could be used to solve the heat equation for thermal conduction in the system. Since the regions of high strain may be distinct from those with high thermal gradients, each grid can be individually tailored to accurately represent the relevant physics.

Now consider the implementation of such a multi-mesh example on a distributed-memory parallel machine. A typical time-step consists of computing a solution on the first mesh, interpolating the result to the second mesh, computing a solution on the second mesh, interpolating it back to the first mesh, and so on. One way of performing this type of computation in parallel is to partition the meshes separately so that every processor has a portion of each mesh. This approach will balance the computations and minimize the communications during each of the solution phases. However, because the different meshes are partitioned independently, there is no assurance that an individual processor will own portions of the meshes that spatially overlap. Therefore, the amount of communication performed during the interpolation and transfer of the solution data can be quite high, even if an efficient approach is used

to manage this communication [7]. Ideally, we would like to partition the different meshes such that each processor performs an equal amount of work for every mesh, and at the same time, the inter-processor communications required during the computations of the solutions, as well as those required during the interpolation and transfer of the solutions, are minimized.

Multi-constraint Graph Partitioning

The common characteristic of these problems is that they all require the computation of partitionings that satisfy more than one balance constraint. Traditional graph partitioning techniques have been designed to balance only a single constraint (i.e., the vertex weight). An extension of the graph partitioning problem formulation is to assign a weight vector of size *m* to each vertex. The problem then becomes that of finding a partitioning that minimizes the inter-processor communication, subject to the constraints that each of the *m* weights is balanced across the subdomains. This *multi-constraint* graph partitioning problem [5] is able to effectively model all of the problems described above.

Figure 6 illustrates an example with three constraints. The graph here is derived from a multi-phase simulation in which each vertex is active during one or more computational phases. The specific phases in which a vertex is active depend upon the region of the graph in which that vertex is located. For example, the vertex in the upper-left corner of Figure 6(a) is in the region of the graph that is active only during the first phase of the computation, while the vertex in the bottom-middle of the graph is active during both the first and second phases. Figure 6(b) shows the weight vectors that are assigned to each vertex. Here, an entry of one indicates that the vertex is active during the corresponding phase and an entry of zero indicates that the vertex is not active during this phase.

We have developed serial and parallel, static and adaptive multi-constraint graph partitioners [5, 8, 10] that are based on this generalized formulation. These have been shown to be effective in computing high-quality partitionings for real applications while simultaneously balancing a number of constraints [1, 8, 11]. Serial versions of these algorithms [5] are included in the widely-used METIS 4.0 graph partitioning library [4]. Parallel formulations of our multi-constraint algorithms [8, 10] have been developed and will be included in the next version of the PARMETIS library [6].

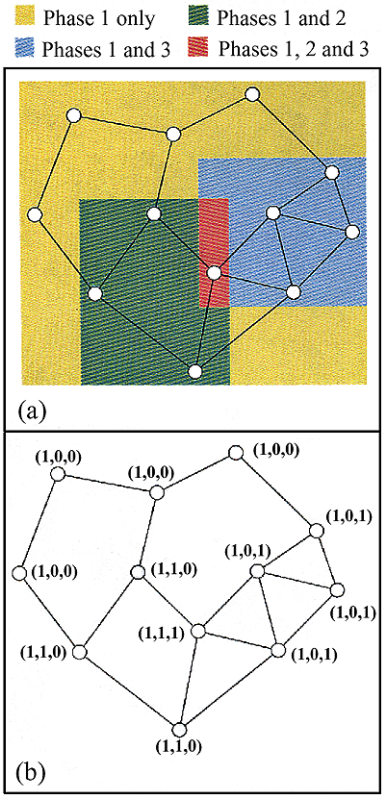


Figure 6: A graph derived from a multi-phase computation.

The shading in (a) indicates the phases in which each vertex is active. The weight vectors in (b) correspond to the active phases.

Author Contact Info

Kirk Schloegel
Dept of Computer Science & Engineering,
University of MN
Army HPC Research Center,
Minneapolis, Minnesota
kirk@cs.umn.edu

George Karypis
Dept of Computer Science & Engineering,
University of Minnesota
Army HPC Research Center,
Minneapolis, MN
karypis@cs.umn.edu

VipinKumar
Dept of Computer Science & Engineering,
University of Minnesota
Army HPC Research Center,
Minneapolis, MN
kumar@cs.umn.edu

References on page 6

continued from page 5

References

1. A. Basermann,J.Fingberg, G. Lonsdale, B. Maerten, and C. Walshaw. Dynamic Multi-partitioning for Parallel Finite Element Applications. *Submitted to ParCo '99*, 1999.
2. B. Hendrickson and R. Leland. The Chaco User's Guide, version 2.0. Technical Report SAND94-2692, Sandia National Laboratories, 1994.
3. B. Jeremic and C. Xenophontos. Application of the p-version of the Finite Element Method to Elasto-plasticity with Localization of Deformation. *Communications in Numerical Methods in Engineering*, 15(12):867-876, 1999.
4. G. Karypis and V. Kumar. MeTiS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, version 4.0. Technical report, University of Minnesota, Dept. of Computer Science and Engineering, 1998.
5. G. Karypis and V. Kumar. Multilevel Algorithms for Multi-constraint Graph Partitioning. In *Proceedings of Supercomputing '98*, 1998.
6. G. Karypis, K. Schloegel, and V. Kumar. ParMeTiS: Parallel Graph Partitioning and Sparse Matrix Ordering Library. Technical report, University of Minnesota, Dept. of Computer Science and Engineering, 1997.
7. S. Plimpton, B. Hendrickson, and J. Stewart. A parallel rendezvous algorithm for interpolation between multiple grids. In *Proceedings of Supercomputing '99*, 1999.
8. K. Schloegel. *Graph Partitioning for Emerging Scientific Simulations*. PhD thesis, University of Minnesota, 1999.
9. K. Schloegel, G. Karypis, and V. Kumar. Graph Partitioning for High Performance Scientific Simulations. In *CRPC Parallel Computing Handbook*. Morgan Kautmann, 2000.
10. K. Schloegel, G. Karypis, and V. Kumar. Parallel Multilevel Algorithms for Multi-constraint Graph Partitioning. In *Proceedings of EuroPar-2000*, 2000. Accepted as a Distinguished Paper.
11. J. Stiller, K. Boryczko, and W. Nagel. A New Approach for Parallel Multigrid Adaption. *Proceedings of the 9th SIAM Conference on Parallel Processing for Scientific Computing*, 1999.
12. C. Walshaw. Parallel Jostle Userguide. Technical Report Userguide Version 1.2.0, University of Greenwich, London, UK, 1998.
13. C. Walshaw. Serial Jostle Library Interface. Technical Report Version 1.2.0, University of Greenwich, London, UK, 2000.

Acknowledgements

This work was supported by DOE contract number LLNL B347881, by NSF grant CCR-9972519, by Army Research Office contract DA/DAAG55-98-1-0441, by Army High Performance Computing Research Center cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. Additional support was provided by the IBM Partnership Award, and by the IBM SUR equipment grant. Access to computing facilities was provided by AHPCRC, Minnesota Supercomputer Institute.

The MeTiS and ParMeTiS libraries are available at www.users.cs.umn.edu/karypis/metis.

An Application of High Performance Computing to Battle Planning

byB. Bodt, J. Forester, C. Hansen, E. Heilman, R. Kaste, J. O'May, U.S. Army Research Lab

Introduction

Command posts in battle are busy hubs for information flow. A commander and his staff must sift through data representing thousands of events to build an accurate picture of a battle. With seconds counting and lives dependent on timely decisions, commanders can use every advantage.

The Simulation Concepts Branch of the U.S. Army Research Laboratory (ARL) is responsible for creation of techniques and systems to assist battlefield commanders in the military decision making process. This process is based in part on Courses of Action (COAs). Our primary focus is COA Analysis (COAA). We are merging operational reality with simulation to benefit the commander and battle staff at Division and lower echelons.

Project Overview

Our project applies military planning and combat simulation software to the evaluation of automated COA generation tools. Envisioned is a testbed that will enable the assessment of COAs in a simulated operational environment. Initially, one COA generation tool, Fox-GA (for Genetic Algorithm) produced by the ARL Federated Laboratory, and one combat simulation, Modular Semi-Automated Forces (ModSAF) produced by Lockheed-Martin, comprise the prototype testbed. The project is exploring statistical analysis and experimental design techniques that will enable simulated exercises to be utilized as a part of COAA.

The project goal is to reveal strengths and weakness inherent in COA generation software. Scientific evaluation of COA attributes will benefit the soldier by providing increased operational possibilities during planning. The complexity of COAA provides fertile ground for development and application of decision aid technology. This work has potential for significantly improving battlefield command and control capabilities.

Methodology

The COAA testbed contains four process parts with associated computer software, hardware, and data. These parts are:

1) Automated COA Generation,
2) Scenario Translation,
3) Experimentation, and
4) Statistical Evaluation.

ARL, in conjunction with theUniversities of Illinois and Minnesota, has created a software tool to generate automated COAs. The program, called Fox-GA, is "an intelligent planning support tool designed to rapidly generate a variety of coarse-grained, high quality, friendly courses of action for military planners."[1] Conversion of the Fox-GA scenario to a form usable by a combat simulation was accomplished manually. Experimentation was accomplished by executing the translated scenario within ModSAF. Data collected from experimental simulation executions were analyzed using variances and means to gauge overall COA performance.

Computing Power Challenge

The Fox-GA scenario portrays combat between a friendly force brigade and an adversary force battalion. The translation captured a complex combat featuring 418 entities. We had difficulty determining the correct amount of computational power necessary to handle this large scenario translation.

Our first attempts to execute the scenario on a Sun Microsystems UltraSPARC60, a system with two 296 MHz UltraSPARC-II

Continued on page 8

continued from page 7

processors, were not successful. In fact, the UltraSparc60 could not complete the scenario setup. Additional systems were added to enable simulation execution in a distributed network environment. The UltraSPARC60 was used to display the plan view and did not support scenario entity activities. Using an incremental approach, five more SGI systems were eventually used to share simulation entities: three SGI O2s with one175 MHz (IP32) R10000 processor each, one SGI Maximum Impact with one 195 MHz (IP29) processor, and one SGI Infinite Reality Onyx with four 194 MHz (IP25) processors. The modified network and hardware configuration was still insufficient to gather data of completed scenarios; all attempts to execute the translated Fox-GA scenario on local computers failed.

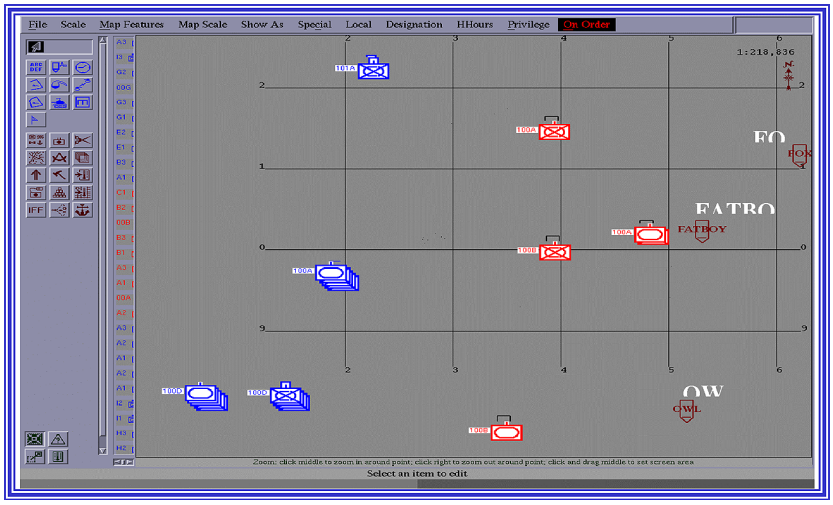


Figure 1: ModSAF Translated Scenario.
Each icon in a stack represents 10 to 14 independent entities.

We used a different ModSAF combat formation to reduce the translated scenario to 280 entities. A lower number of entities resulted in the generation of fewer network packets. Yet the networked solution was still insufficient for ModSAF data collection. A network packet bundling technique was used to diminish the number of network transmissions, but also did not provide enough network capacity to alleviate the lack of computing power.

We decided that super computers would be needed to enable data collection from combat simulation runs. The entire process, from obtaining accounts on ARL's Major Shared Resource Center's high performance computers to actual scenario execution, was accomplished in the space of five days. We installed the ModSAF software on four SGI Origin 2000 systems with little difficulty. Each system has at least 32 processors (250 MHz IP27 R10000) and 32 gigabytes of main memory. These features enabled a single SGI Origin 2000 system to support successful scenario completion and data collection, while eliminating the requirement for network transmissions.

Conclusion

Without the availability of super computers, the resources available for this project would have been insufficient to provide results. The simulation data compiled using one Origin 2000 system over the space of two weeks show that ModSAF and Fox-GA are consistent, as friendly forces were winning most of the time in each. Fox's internal wargamer may be good enough to provide battle results for competing COAs; however further experimentation will be necessary to gauge the accuracy and degree of these results. Battlefield commanders will benefit from laboratory-tested, expedient decision aids, and we will continue to use the power of super computers to improve the Army's command and control abilities.

Authors Contact Information

B. Bodt, J. Forester, C. Hansen, E. Heilman, Richard Kaste, J. O'May

Each of the authors may be reached by contacting:
Director, U.S. Army Research Laboratory
Attn: AMSRL-CI-CD
Aberdeen Proving Ground, MD 21005

(410) 278-7778, Fax 410.278.4988

rck@arl.army.mil (Kaste)

References

1. C. B. Fiebig-Brodie, C.C. Hayes, "Evaluating The Utility of Decision Support Tools to Assist in Army Tasks," Advanced Displays & Interactive Displays Consortium, Proceedings of the ARL Federated Laboratory Symposium 2000, 21 – 23 March 2000, 37.

Bibliography

Fiebig, C. B., C. C. Hayes, and R. P. Winkler. "What's New In Fox-GA." Advanced Displays & Interactive Displays Consortium, 3rd Annual FedLab Symposium, 2 - 4 February 1999, 9.

Lockheed-Martin. MODSAF 5.0 Software Architecture Design and Overview Document, 3.

Marshak, William P., Carolyn Fiebig, Robert Winkler, Robert Stein, and Albert Khakshour. "Evaluating Intelligent Aiding Of Course Of Action Decisions Using The Fox Genetic Algorithm in 2-D and 3-D Interfaces." Advanced Displays & Interactive Displays Consortium, 3rd Annual FedLab Symposium, 2 - 4 February 1999, 27.

Schlabach, J. L. and C. C. Hayes. "Fox-GA: A Genetic Algorithm For Generating and Analyzing Courses Of Action." Advanced Displays & Interactive Displays Consortium, ARL Federated Laboratory Annual Symposium, 1998, 40.

Uckun, Serdar, Selim Tuvi, Rex Winterbottom, and Patrick Donohue. "OWL: A Decision-Analytic Wargaming Tool." Advanced Displays & Interactive Displays Consortium, 3rd Annual FedLab Symposium, 2 - 4 February 1999, 133.

Announcement: DTIC 2000
Defense Technical Information Center (DTIC) Annual Users Meeting and Training Conference

"Information Solutions for the 21st Century"

The Defense Technical Information Center (DTIC) will host DTIC 2000, its Annual Users Meeting and Training Conference, from 6-9 November 2000 at the Double Tree Hotel Rockville, Maryland.

Conference Theme and Objective

This year's theme, "Information Solutions for the 21st Century," reflects DTIC's primary objective: to assist its customer community in meeting tomorrow's challenges by providing the most relevant information in the most appropriate format as quickly as possible.

DTIC 2000 provides a unique opportunity for attendees to explore in detail new developments not only at DTIC, but throughout the federal technical information network. As in past years, the conference will feature a number of presentations and sessions that focus on the most current issues relative to the research, development and acquisition communities.

These sessions are designed to acquaint the participants with the latest policy and operational developments, and will provide practical details on valuable and diverse domestic and foreign information resources. They will also address security issues, the World Wide Web, copyright laws and the storage and dissemination of electronic documents.

"Information Solutions for the 21st Century" will provide timely, accurate information which will enable users to better meet the challenges of the future. It also promises to provide the tools needed to expand participants' horizons to meet these challenges.

DTIC 2000

6-9 November 2000

Double Tree Hotel
Rockville, MD

Registration

For more information, please contact:

Ms. Julia Foscue
DTIC 2000 Conference Coordinator
(703) 767-8236
jfoscue@dtic.mil

Access the latest information on the DTIC Website:
http://www.dtic.mil

Architecture and Performance of a Parallel Data-Fusion Multi-Target Tracking Code for Real-Time Applications

by Dr. Charles R. Pedersen, Visiting Scientist, AFRL, Rome Research Site

Abstract

An existing serial code for fusing asynchronous data from multiple radar and optical sensors, and then simultaneously initiating, maintaining and dropping tracks on multiple missile targets, has been successfully parallelized at the Air Force Research Laboratory (AFRL) under the aegis of the DoD High Performance Computing Modernization Office (HPCMO).

This article is taken from a paper which describes the functionality of the original serial code and the implementation of a parallel software architecture that distributes these functions across the multi-processor parallel architecture of the HPCMO Intel Paragon Computer at AFRL Rome Research Site (RRS). The objective of the work is to demonstrate a Fusion Tracker code that is both parallel and portable, potentially for implementation on real-time embeddable High Performance Computing (HPC) architectures. The paper is available in its entirety on the DACS Website at: www.dacs.dtic.mil/awareness/newsletters/stn4-1/datafusion.html

The improved performance levels achieved by the Parallel Fusion Tracker are presented for the main metrics of interest in real-time applications, namely latency, total computation load, and total sustainable throughput. Results are presented for combinations of 1 to 126 targets being tracked on 1 to 126 parallel nodes, up to a total of 126 targets per node. It is shown that the single, key parameter that determines both latency and overall throughput is the number of targets per node, irrespective of other parameter variations. It is further shown that overheads introduced by use of the Message Passing Interface (MPI) have negligible significance when there are greater than 4 to 8 targets per node. Code and algorithm optimization of the baseline serial code remain as future tasks for achieving higher parallel performance levels above the 115 to 257 MFLOPS reported here.

Introduction

When a digital computer is contemplated for application in a real-time signal or information processing environment, an issue of major concern is "latency," the time delay between data being received at the computer input and the effects of that data appearing at its output. Latency, or information timeliness, in any given application is determined both by the amount of computation to be done and the computational speed of the computer doing it, i.e. its effective "throughput," measured in millions of floating operations per second (MFLOPS). As the computational requirements for future real-time defense applications grow, seemingly without bound, the need to maintain information timeliness can be addressed by: 1) optimizing the processing algorithms; 2) increasing computational speed of the processor; or 3) applying many processors in parallel High Performance Computing architectures. In fact, all three of these technology avenues are continually being pursued, and when they are used in combination they lead to the record-breaking computational landmarks of the day. This paper specifically addresses the third alternative for performance improvement, namely parallel processing.

This paper is a study of the application of parallel high performance computing to a candidate serial algorithm for jointly accomplishing data fusion from many sensors and simultaneously tracking multiple targets in real-time. The emphasis is on comparing the architectures of the serial and parallel algorithms, and characterizing the performance benefits achieved by the

continued on page 12

Parallel Data-Fusion Multi-Target Tracking Code

continued from page 11

parallel algorithm. In addition to technical results this paper includes a discussion of the parallelization effort and particular lessons learned from it.

System Context

The future battlespace scenario that motivates the Fusion Tracker code is illustrated in Figure 1.

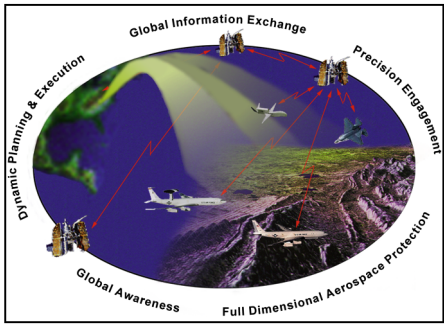


Figure 1: Notion Planning Scenario

The notional scenario includes multiple satellite, airborne and ground radar, optical and infrared surveillance sensors viewing a large geographic footprint that contains large numbers of friendly, hostile and other targets of various types, whether aircraft, missiles or ground vehicles. In such a situation the amount of information that can be collected, especially when imaging is included, is simply enormous and will swamp human interpretation and reaction times.

A simplified block diagram that describes the context for the data fusion and tracking program to be discussed is shown in Figure 2. It has been assumed that, prior to arriving at the fusion tracking function, the data collected by a number of sensors individually has been reduced to individual target detections. Each detection arrives at the fusion tracker input with associated target coordinates and data quality flags. In development of the baseline serial code it was further assumed that the results of data fusion and tracking are collected into a data server that contains the state vectors for each individual target in the battlespace. In this design, the downstream functions of target identification, prioritization, scheduling, interdiction resource allocation and so forth can then function asynchronously from the fusion and tracking process, and can draw necessary target information from the state vector server as needed.

The functions that then fall to the multi-sensor data fusion and multi-target tracking code include Input Measurement Processing, Data Association and Track Maintenance, and State Vector Management, as shown in Figure 2.

There were two driving requirements for the Parallel Fusion Tracker code: first, that detections and metric coordinates from 100 targets and other objects simultaneously in the multiple fields of view would have to be processed in real time; and second, that embeddable HPC technology would be used in order to support HPC deployments in mobile and airborne applications.

Objective

The objective of this code parallelization effort was, therefore, to demonstrate an implementation of multi-sensor data fusion and multi-target tracking functions within an integrated multi-node portable HPC architecture. The key metrics to be determined in support of ongoing system analyses included: required computational throughput in MFLOPS; latency between receipt of input data and resulting outputs; and scalability, processor utilization and memory requirements. Furthermore, the standard Message Passing Interface (MPI) functions were to be used for inter-node communications in order to promote code portability across multiple HPC computer platforms.

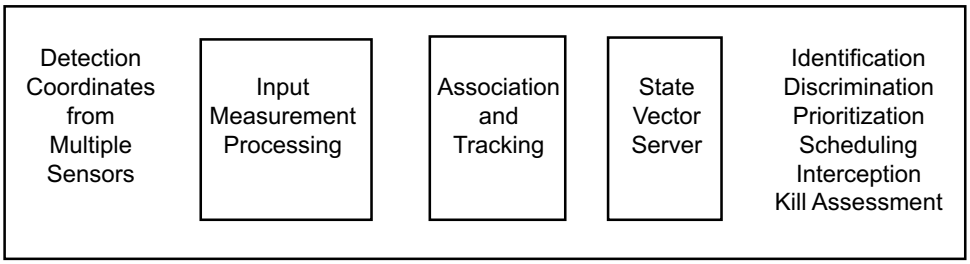


Figure 2: System Context for Parallel Fusion Tracker Functions

Conclusions

A major factor in determining the measured performance levels is the fact that the fusion tracker code was parallelized "as-is" without optimization of either the C code or the algorithms employed. Both the serial and parallel fusion tracking codes are written in straight C code (matrix routines included), plus MPI calls for the parallel version. Neither uses library functions optimized for the Paragon nor do they otherwise engage in cache management to speed the calculations.

In fact each of these considerations applies equally well to both serial and parallel code and the standard advice still remains true. Truly high performance begins with selection of a good serial algorithm, followed by optimization of it, followed then by parallelization. In the case of the Parallel Fusion Tracker, code improvement and optimization remain as tasks for the future.

A serial data fusion and tracking code has been successfully parallelized through the use of standard MPI communication functions. Apart from debugging, the two major components of effort were 1) reverse engineering of the original code in the absence of detailed code documentation, and 2) creation, declaration and definition of portable MPI derived data types to correspond to each of the complicated data structures present in the original code. The result is a Parallel Fusion Tracker code written in standard C and MPI that will be portable to other High Performance Computing architectures, such as the 384-node Power-PC-based Sky computer recently acquired by AFRL Rome Research Site.

The Parallel Fusion Tracker was instrumented and detailed data was collected for the latency and computational effort contributed by each of its 15 major functional modules, for numbers of targets and nodes between 1 and 126, in combinations ranging between 1 and 126 targets per node. The relationship between total computational load, represented by number of targets being tracked, and the resulting latency and throughput MFLOP levels were determined as functions of the number of nodes. Above 4 to 8 targets per node it was shown that the code is highly scalable, meaning that latency and number of nodes can be traded against each other at any total computational level. Latency, for example, might therefore be held to a particular value by choosing the number of nodes appropriate to the number of targets to be tracked. The governing functions for both latency and computation per node are shown to be very nearly deterministic functions of just one single variable, namely targets per node, over very wide ranges of numbers of targets or of nodes.

Throughputs for individual functions as high as 6 MFLOPS were recorded for Track Propagation, as well as 257 MFLOPS for the integrated Association and Tracking function of the overall Fusion Tracker when tracking 126 targets. When Input Processing and State Vector Management are included they contribute more to latency than they do to computation, and overall performance levels are diluted to 5 MFLOPS and 115 MFLOPS respectively when tracking 126 targets on 126 nodes. These levels, though low for parallel computation, are nevertheless typical for codes that simply rely on straightforward C coding without the use of optimized library functions or careful memory management during computation. Code optimization of the baseline serial code and its parallel version remain as tasks for the future.

About the Author

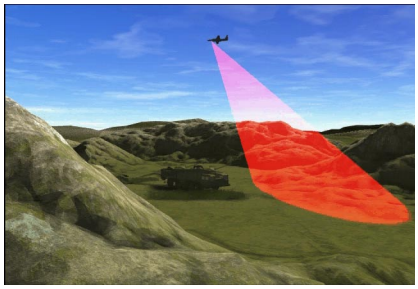
Charles R. Pedersen, Ph.D
AFRL, Rome Research Site
26 Electronic Parkway,
Rome, NY, 13441
(315) 330-4846
pedersen@rl.af.mil

AFRL Sensors Directorate, a One-Stop Shopping Resource for Automatic Target Recognition (ATR)

by Capt. Paul Harmer, Wright Patterson Air Force Base

Wright Patterson Air Force Base, Ohio (AFRL/SN) – The mission of the Air Force Research Laboratory (AFRL) Sensors Directorate is to ensure unequaled reconnaissance, surveillance, precision engagement and electronic warfare capabilities for America's air and space forces, by conceiving, demonstrating and transitioning advanced sensors and sensor technologies. In simple terms, we're the "*Eyes and Ears of the Warfighter.*" We accomplish this daunting task in partnership with industry, universities and other DoD agencies.

A recent announcement by the Department of Defense High-Performance Computing Modernization Office (HPCMO) provides the Sensors Directorate with one more tool to support our mission. The HPCMO announced that AFRL/SN is now an HPC Distributed Center and awarded the funds needed to purchase needed HPC resources. Acquiring HPC resources at AFRL/SN adds a much-needed capability to an ongoing program sponsored by the Office of Secretary of Defense (OSD) called the Virtual Distributed Laboratory (VDL).



The VDL is an information distribution center that provides researchers the capability to talk about, share, store, search, and retrieve information related to Automatic Target Recognition (ATR) and Information Fusion technologies. The addition of HPC provides users with the computing power needed to process these technologies in real time. Currently, VDL supports 31 programs and over 800 users, including the Air Force, Army, Navy, DARPA and several other DoD level organizations. A more detailed explanation about VDL will be provided, but first we need to understand the goals of ATR and sensor fusion.

Remember, the goal of the Sensors Directorate is to be the "*Eyes and Ears of the Warfighter.*"

So, one question to ask is: **"What does the warfighter (in this example, battlefield commander) need to see and hear?"**

If you have ever played the childhood game of Hide and Seek, you understand two of the basic needs of a battlefield commander. First, the commander has to find where the enemy is hiding. Next, he has to use that information to formulate a plan of attack. In hide and seek terms, he has to find where the people are hiding and tag them before they reach home base and are safe. In the game, the people you are trying to find have the advantage because the seeker has to hide his/her eyes and count to some number while the people hide. Similarly, in warfare, if the enemy can move around and hide while we can't see him, he has the advantage. Our goal in the Sensors Directorate is to give the battlefield commander the ability to look while the enemy is trying to hide or to find him after he has hidden.

One way we do that is by developing and fielding remote sensors that can "see" and "watch" over an area of interest and provide that information to the commander in the form of images. Image analysts review these images to decide where the enemy is hiding and what he may be planning. A problem arises when you have too much image data and not enough image analysts to review it. Using the hide and seek example, it's like being able to look while you count, but having to count to a million before seeking. In the game, the players who are hiding are likely to quit and go home before you finish counting to a million and begin to look for them. In war, if it takes too long to analyze and react to

data, the information becomes useless. Therefore, the ability to see what the enemy is doing without the ability to analyze and to act on this information in a timely manner is useless. ATR technologies are designed to make timely analysis of massive amounts of sensor data possible.

Researchers design ATR technologies that automatically process remote sensor data using computer algorithms to extract, sort, condense, and fuse the relevant information in these images. The processed information is then presented to battlefield commanders, allowing them to make faster and better battlefield decisions. Therefore, the commanders not only get the ability to look while counting, but they also get the relevant information in a timely manner. This allows them to react while the information is still useful. The question is: How does VDL help to accomplish this?

VDL is a high-speed high bandwidth interactive network system that facilitates collaboration between nationally distributed DoD research and developers using the following five functional areas:

(1) HPC Distributed Center

(2) Central Information Library

(3) Distributed Data Query Tools

(4) Collaboration Tools

(5) Search Engines

In addition, it allows honest-broker evaluation of ATR algorithms and data sets. In simple terms, VDL helps researchers and developers from all over the country work together to develop better ATR and Information Fusion technologies. With the addition of embedded HPC resources, it will also help to reduce the time it takes to get those technologies to the warfighter. Finally, combining VDL with a high-speed network foundation provides worldwide users with the very best in remote information management and high performance computing to directly support Information Superiority in the 21st Century. One of the most critical steps to getting technology to the warfighter is making sure it can be fielded. Which means, it must fit in an aircraft platform, in a mobile ground unit, or on a ship. It also must support the shock, temperature, and power requirements that go along with those environments. Because of size, weight, shock, temperature, and power requirements, our HPC Distributed Center (DC) will use embedded HPC resources. Embedded High Performance Computers provide the CPU power needed to support the real-time requirements for Automatic Target Recognition and Information Fusion, while at the same time support the fielding requirements.

As stated, in future conflicts it will not be possible or adequate to throw more people at the problem. Battlefield awareness will require the acquisition, assimilation, and analysis of vast amounts of remote sensor data at rates that far exceed the capabilities of human analysts. ATR technologies can bridge that gap, directly supporting "Information Superiority." A distributed center dedicated to the real-time requirements for Automatic Target Recognition (ATR), signal/image processing (SIP), and Integrated Modeling and Test (IMT) can demonstrate greater than a 10x speedup in development and transition of critical ATR technologies to warfighters.

continued on page 16

continued from page 15

Conclusion

HPC resource centers allow technologies to be developed using similar, and in some cases, the same hardware currently deployed in the field. The VDL HPC center provides a capability that greatly reduces the cost of ATR research and development by lowering and eliminating duplication of efforts in data collection and retrieval, information distribution and data storage. It also provides a cost avoidance to organizations from engineers and scientists' time spent conducting ATR, Fusion, and C4IRS literature searches that often result in no matching, overwhelming matches that necessitate detail reading of advertisements, and unrelated subject matter when using the Internet super highway to information.

The VDL HPC center emphasizes that the center cannot store everything of interest to the DoD community. A huge amount of additional information is stored on other program, government, academic, and industry web sites. Commercial search engines are great for locating such distributed information. "Our problem has been that the public search engines index too much content for our purposes" stated by the Center's team leader, Captain Paul Harmer. The VDL Central Library provides researchers the capability to quickly find pertinent information, and spend more time developing solutions and solving problems. This capability will result in providing faster technology transition to the war-fighters, and improve the effectiveness and efficiency of technical solutions to complex and recurring weapon system issues.

Virtual Distribute Laboratory (VDL)

www.vdl.afrl.af.mil/

The addition of an interactive Real-Time High Performance Distributed Center to the VDL program provides the capability to develop and share high-performance interactive modeling and simulation tools and to create and evaluate advanced embedded ATR technologies. It also provides a development infrastructure that mirrors many of the embedded system hardware architectures currently in the field. Eliminating the need to port code or at least making it much easier facilitates rapid demonstration/transition of new technology to warfighters.

The bottom line is that the addition of a real-time classified and unclassified interactive HPC, for shared use by the DoD community, affords a unique opportunity to develop and share ATR technologies across the DoD and enables successful transition of HPC technology to the fielded production systems.

Author Contact Info

Captain Paul Harmer
AFRL/SNAS, Bldg. 23
Fifth St.
WPAFB OH 45433-7001

(937) 255-6329 DSN: 785-6329
Fax: (937) 255-1100 DSN: 255-1100

Public Release Number: ASC2000-605

An Overview of AFRL HPC Efforts

Continued from page 1

The configuration being purchased by AFRL includes 384 processors in three chassis and is capable of computer performance of 640 GFLOPS. This SKYchannel System is modular and can easily be reconfigured into three separate computers for phased deployment to the field, with a maximum of 554 GFLOPS for a single chassis.

The compute power of AFRL's new system is derived from 24 PowerPC-based SKYchannel 9U boards combined with the SKYchannel crossbar switching fabric and system software to deliver excellent performance on key benchmarks for signal and image processing. This SKYchannel system is a general-purpose system with a distributed memory architecture.

The design of the memory, communication, and I/O subsystems was optimized for operating on vectors and matrices of data, such as those found in signal and image processing applications.

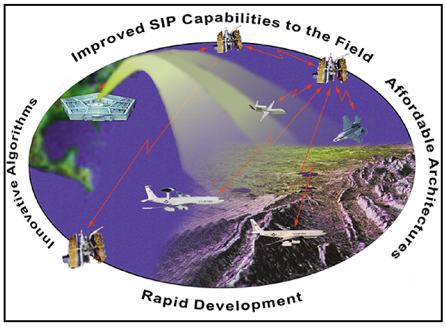
The SKYchannel System includes system software, development tools, and run-time libraries to support real-time signal and image processing. Foremost among these are advanced compilers and libraries that automatically vectorize application software without depending on code or function parameters tuned for a particular processor or system architecture. With no requirement to learn the processor architecture and then code that into the application, an applications developer can focus on the algorithms while reducing the time to develop, debug, and field test the system.

AFRL accepted and began providing tri-service access to one chassis of the system in June 2000. AFRL is currently testing a second chassis configured with nine boards full of PPC 7400 processors with a peak rating in excess of 450 GFLOPS. This chassis is expected to be ready for development and field testing by the end of September 2000. The third chassis is expected to remain at AFRL to support development activities leading up to field tests. Boards can be easily moved between chassis to support experimental requirements. DoD researchers are encouraged to get accounts and experiment with the system. Contact Virginia Ross for system access and test information and to discuss opportunities for field tests.

Author Contact Info

Virginia Ross
Air Force Research Laboratory/IFTC
26 Electronic Parkway
Rome, NY 13441
(315) 330-4384
Fax: (315) 330-2953

virginia.ross@rl.af.mil



System Performance	
Peak Floating-Point Performance	640 GFLOPS
Peak Bisectional Bandwidth	1.92 GBytes/sec
Total System Memory	26 GBytes

System Configuration Flexibility

Each of three systems racks operate independently, and may be physically disconnected from the others. Any or all three 9U SKYchannel chassis may be removed from the systems racks and deployed independently.

continued on page 18

An Overview of AFRL HPC Efforts

Continued from page 17

System Configuration	
Signal Processors	(144) PowerPC 7400 and (240) PowerPC 604e – 333 MHz processors, each running an 83.3 MHz memory interface
I/O Processors	(24) Intel i960CA processors
Signal Processor Boards	(24) 9U VME SKYchannel boards, each with (16) Signal Processors, (1) I/O processor, 1 GByte memory, (1) FPDP parallel port, and (1) SKYchannel communication port
System Area Network	(8) SKYchannel Backplanes and (2) SKYchannel Chassis Hubs, each using SKYchannel Crossbars for multiple 320 MB/sec connections
Additional I/O Interfaces	(9) Fibre Channel Interfaces (4) HIPPI Interfaces (1) ATM-OC3 Interface
Processor Chassis	(3) 9U SKYchannel Chassis,each with (8) Signal Processor boards, (1) UltraSPARC host processor board, and standard peripherals (CD, HD, Tape)
RAID Systems	(3) 50 GByte RAID level 5 subsystems (1) 144 GByte RAID level 5 subsystem
System Racks	(3) System Racks, each with (1) 9U SKYchannel Chassis, (1) RAID System. Two racks have (1) SKYchannel Chassis Hub each.
System Expandability	System may be expanded without changing the configuration by adding processor boards to spare slots, up to 16 SKYchannel boards per chassis, and/or by adding another processor chassis.

Software Development and Run-Time System	
Host Operating System	Solaris 2.7
Compilers	Automatic Vectorizing SKYvec C/C++ and SKYvec Fortran
Signal Processing OS	SKYmpx Real-Time OS
Communications Libraries	SKYscl Scalable Communications Library MPI/RT
Signal Processing Libraries	SKYvec SML VSIP (Vector, Signal, and Image Processing) Library
Real-Time Development	TimeScan Multiprocessor Event Analyzer

Software Tech News on the World Wide Web

This newsletter, including referenced full-length articles, is available on the web at:
www.dacs.dtic.mil/awareness/newsletters/

High Performance Computing On-line Resources

DACS High Performance Computing Topic Area - www.dacs.dtic.mil

Advanced Scientific and Engineering Computational Center (ASECC) - www.npt.nuwc.navy.mil/asecc/

Army High Performance Computing (HPC) Research Center - www.arc.umn.edu

Center for Computational Science (CCS) - www.nrl.navy.mil/CCS/

High Performance Computing Modernization Program - www.hpcmo.hpc.mil

HPC Wire (On-line Magazine for HPC) - www.tgc.com/hpcwire.html

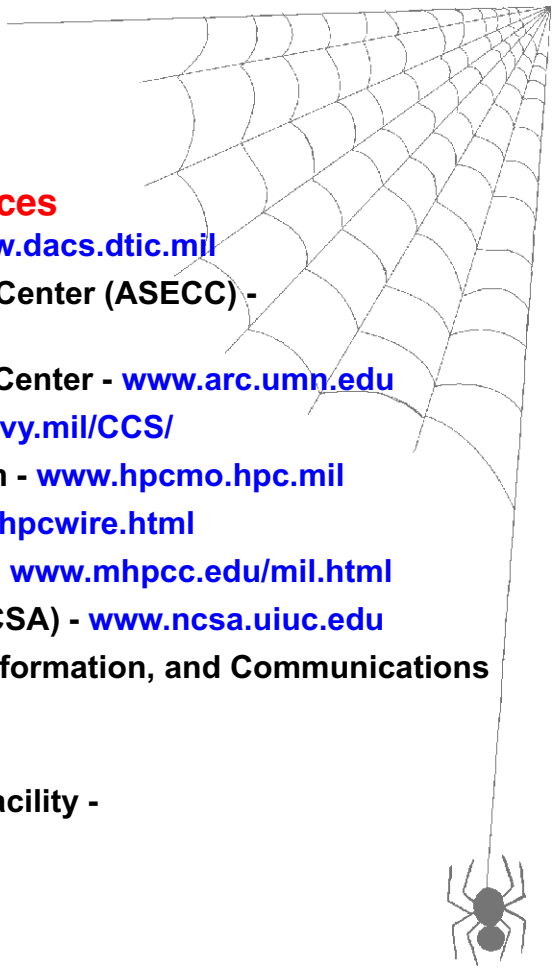
Maui High Performance Computing Center (MHPCC) - www.mhpcc.edu/mil.html

National Center for Supercomputing Applications (NCSA) - www.ncsa.uiuc.edu

National Coordination Office (NCO) for Computing, Information, and Communications (CIC) - www.hpcc.gov

National HPCC Software Exchange - www.nhse.org

Rome Research Site High Performance Computing Facility - www.if.afrl.af.mil/div/IFT/IFTC/RLHPC.html



Article Reproduction

Images and information presented in these articles may be reproduced as long as the following message is noted:

> "This article was originally printed in the DACS *DoD Software Tech News*, vol. 4, no. 1.
> Requests for copies of the referenced newsletter may be submitted to the following address:
>
> Data & Analysis Center for Software
> Attn: Lon R. Dean, Editor
> PO Box 1400
> Rome, NY 13442-1400
> (800) 214-7921; Fax (315) 334-4964;
> news-editor@dacs.dtic.mil
>
> An archive of past newsletters is available at www.dacs.dtic.mil/awareness/newsletters/.

In addition to the copyright message, we ask that you send us three copies of any document that references any article appearing in the *DoD Software Tech News*.

Thank you for your interest in the products and services of the Data & Analysis Center for Software (DACS).

Data & Analysis Center for Software
P.O. Box 1400
Rome, NY 13442-1400

DACS Products & Services Order Form

Name: Position/Title:

Organization: Acronym:

Address:

City: State: Zip Code:

Country: E-mail:

Telephone: Fax:

Product Description	*Format*	*Quantity*	*Price*	*Total*
The DACS Information Package	***Note: All Disks are available in PC or Mac**			
❑ Including: 2 recent Software Tech News newsletters, and several DACS Products & Services Brochures	Documents		FREE	FREE
Empirical Data				
❑ Architecture Research Facility (ARF) Error Dataset	Disk		$ 50	
❑ Goel-Okumoto Software Reliability Model 1.0				
❑ NASA / Software Engineering Laboratory (SEL) Dataset	CD-ROM		$ 50	
❑ NASA / AMES Error/Fault Dataset	Disk		$ 50	
❑ Software Reliability Dataset	Disk		$ 50	
❑ DACS Productivity Dataset	Disk		$ 50	
Technical Reports		*FREE with Spreadsheet* →		
❑ A Business Case for Software Process Improvement (Revised)	Document		$ 25	
❑ Measuring ROI from Software Eng Management Spreadsheet	CD-ROM		$ 50	
❑ A History of Software Measurement at Rome Laboratory	Document		$ 25	
❑ An Analysis of Two Formal Methods: VDM and Z	Document		$ 25	
❑ An Overview of Object-Oriented Design	Document		$ 25	
❑ Artificial Neural Networks Technology	Document		$ 25	
❑ A Review of Formal Methods	Document		$ 25	
❑ A Review of Non-Ada to Ada Conversion	Document		$ 25	
❑ Software Design Methods	Document		$ 25	
❑ Distributable Database Technology	Document		$ 25	
❑ Mining Software Engineering Data: A Survey	Document		$ 50	
❑ Object Oriented Database Management Systems (Revisited)	Document		$ 50	
❑ Software Analysis and Testing Technologies	Document		$ 25	
❑ Software Design Methods	Document		$ 25	
❑ Software Prototyping and Requirements Engineering	Document		$ 25	
❑ Software Interoperability	Document		$ 25	
❑ Software Reusability	Document		$ 25	
❑ Understanding & Improving Technology Transfer in Soft Eng	Document		$ 50	
❑ Using Defect Tracking & Analysis to Improve Software Qual	Document		$ 50	
❑ DACS Technical Reports CD (Includes reports listed above)	CD-ROM		$ 250	
Bibliographic Products				
❑ Rome Laboratory Research in Software Measurement	Document		$ 25	
❑ DACS Custom Bibliographic Search	Disk		$ 40	
❑ DACS Software Engineering Bibliographic Database (SEBD)	CD-ROM		$ 50	

Number of Items Ordered ☐ *Total Cost* ☐

Method of Payment:

❑ Check ❑ Mastercard ❑ Visa

Credit Card # _____ Expiration Date _____

Name on Credit Card _____ Signature _____

Mail this form or: Phone: (800) 214-7921, Fax: (315) 334-4964
E-mail: **cust-liasn@dacs.dtic.mil**

This form is also on-line at: www.dacs.dtic.mil/forms/orderform.shtml

---fold here---

---fold here---

_____

_____

_____

Fix
postage
here

Data & Analysis Center for Software
Attn: DACS Customer Liaison
PO. Box 1400
Rome, NY 13442-1400